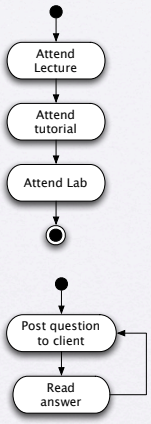
Activity diagrams

Activity diagrams

- Allow you to model complex behaviour/workflows
- Similar to flowcharts except they allow parallel behaviour
- Each state in the diagram is an activity state (where some action occurs)
- Ref. UML Distilled, Martin Fowler



Why activity diagrams?

- Model business workflows
- Identify candidate use cases through examination of workflows
- Identify pre/post conditions for use cases
- Model workflows between use cases
- Model workflows within use cases
- Model complicated workflows in operations on objects
- Model lower level details of complex activity diagrams

Ref. Schaum's Outlines UML, Bennet, Skelton, Lunn

Activity diagrams

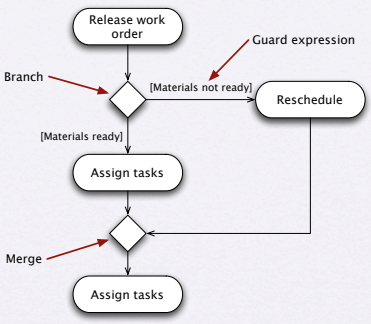
- Activity diagrams describe the flow of activities to achieve a goal/ user requirement.
- They are an easy form to understand - clients will have no trouble reading them.
- Good to identify activities that can take place in parallel
  - Many systems simply replicate existing people-based systems without thinking about what needs to be done sequentially and what can be done in parallel.

Activity diagrams

- Consist of
  - Activities
    - exited when the activity is complete
  - Transitions
    - a path from one activity to another
  - Synchronisation bars
    - ensures transitions happen in unison
  - Decision diamonds
    - if statements…
  - Start and stop markers
    - the boundaries of the activity diagram

Branching

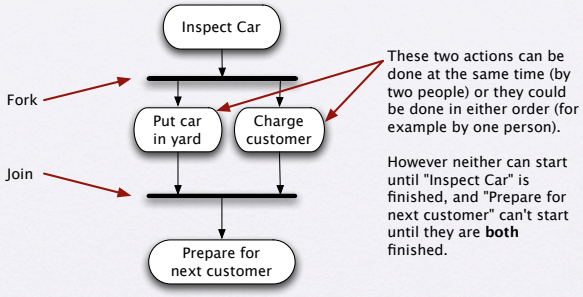
- Allow for different paths in a flow
- Guards are boolean conditions that
  - must not overlap (otherwise the diagram would be ambiguous)
  - must cover all possibilities
- A branch can have more than two leaving paths
- A branch can be labelled "else" to catch any other condition



Ref: The Unified Modelling Language Guide, p273, 2nd edition

Forking and joining

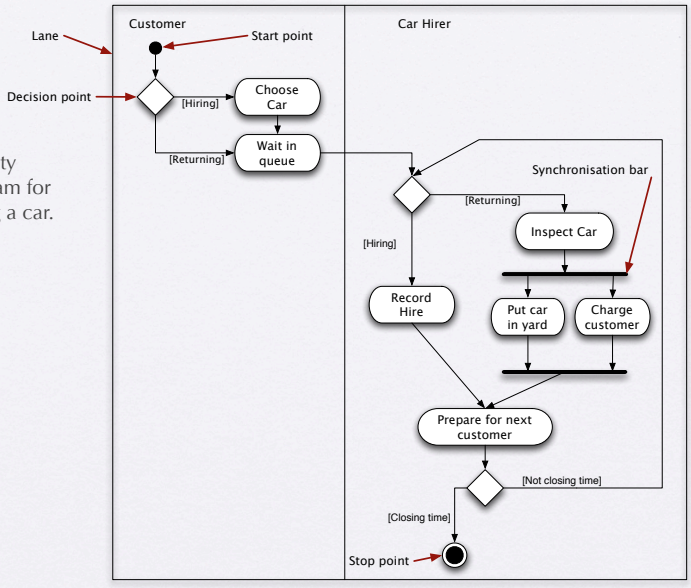
- Synchronisation bars reduce the ordering on activities. Without them all activities would have to be performed sequentially.



Swim(ming) lanes

- The activity diagrams so far show the actions within a workflow, but not who does them.
- We can show who is responsible for the actions by using "swimlanes" - all the actions within one are done by one entity/role
- Swimlanes *doesn't have to* = class or actor
- Swimlanes are any grouping you think makes sense

- Activity diagram for hiring a car.



Problem

- Draw an activity diagram for enrolling in university
  - The form you fill in is validated by a staff member - if a mistake is found you should correct and resubmit it.
  - You can attend a welcome session before or after you deal with payment issues
- Does this cross more than one use case?
- When would you create this?

Problem

- Create an activity diagram for this use case.

Name Place bid
Description A student places a bid on a listed item.
Actors /Bidder
Normal course
1. Student views the recommended book lists for the courses they are enrolled in.
2. Student selects a book from this list.
3. Student views a list of items that are listed for that book, along with their current bid amounts.
4. Student selects an item from this list.
5. Student enters a bid amount greater than the current bid.

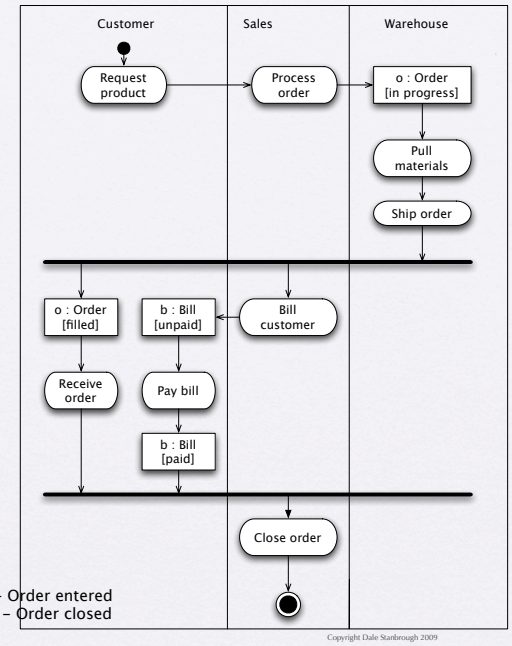
Alternative courses

Student is not enrolled in any courses. Replace step 1 with:
• A message is displayed indicating that the student is not enrolled in any courses, and cannot bid on items.

No items are listed for the selected book. Replace step 3 with:
• A message is displayed indicating that no items are listed for that book, and recommending that the student purchase the book from the RMIT bookshop.

Object flows

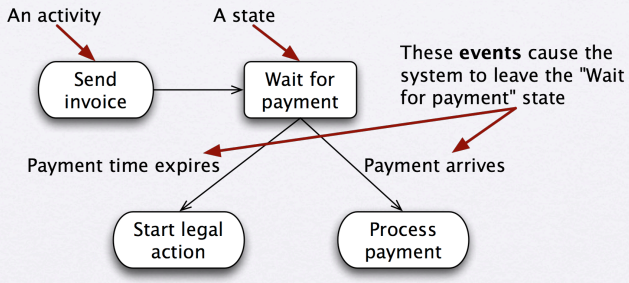
- Objects may be involved in the flow of control in an AD.
- We can show objects being created, change of state etc.
- There is a mistake in this diagram. What is it?
- Can you see a link to state diagrams in these objects?



<<precondition>> – Order entered
<<postcondition>> – Order closed

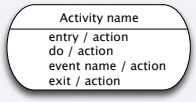
States

- Activities can be combined with states (a state is a condition where we are waiting for something to happen).
- Generally an activity is where we something happens, whereas a state is where nothing happens.

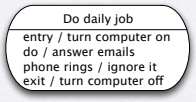


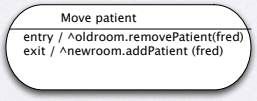
Behaviours within Activities

- An activity can specify behaviours that are triggered
  - on entry
  - while in the activity
  - when an event occurs
  - on exit



- You can write these in plain language, or in a more structured format
  - ^target.event (arguments)
  - e.g. ^room.addPatient (fred)





Activity diagrams

- An activity can be turned into a composite activity if we need more detail.